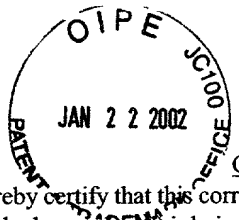
#4

COPY OF PAPERS
ORIGINALLY FILED

PATENT



01SW100

CERTIFICATE OF MAILING

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231

Date: 11-13-01

Himanshu S. Amin

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicants: McKelvey, *et al.*

Serial No: 09/928,623

Filing Date: August 13, 2001

Examiner: Unknown

Art Unit: Unknown

Title: INDUSTRIAL CONTROLLER AUTOMATION INTERFACE

Assistant Commissioner for Patents
U.S. Patent and Trademark Office
Washington, D.C. 20231

PRELIMINARY AMENDMENT

Dear Sir:

Entry of this preliminary amendment is respectfully requested prior to performing substantive examination of the above-identified patent application.

CLEAN VERSION OF REPLACEMENT PARAGRAPHS TO THE SPECIFICATION

Replacement paragraphs to the specification are shown in this section for purposes of clarity. The marked up version of the specification is shown at pages 8-11 of this Reply.

Please replace the 3rd paragraph (lines 7-9) on page 4 with the following:

Fig. 5 illustrates a flow diagram of a methodology for downloading a control program to a processor of an industrial controller through the automation interface in accordance with one aspect of the present invention.

Please replace the 4th paragraph (lines 10-12) on page 4 with the following:

Fig. 6 illustrates a flow diagram of a methodology for uploading a control program to a processor of an industrial controller through the automation interface in accordance with one aspect of the present invention.

Please replace the 5th paragraph (lines 13-15) on page 4 with the following:

Fig. 7 illustrates a flow diagram of a methodology for inserting a rung into a control program and downloading the control program to a processor of an industrial controller through the automation interface in accordance with one aspect of the present invention.

Please replace the 4th paragraph beginning on page 7 (lines 31-32) and ending on page 8 (lines 1-10) with the following:

For example, if the automation interface 14 includes compiled COM libraries, the client application can access the automation interface through a local procedure call (LPC) or a remote procedure call (RPC). A set of proxies and stubs (DLLs) are provided to marshall and unmarshall parameters associated with local and remote calls. The automation interface 14 is provided with a set of classes (*e.g.*, C++, JAVA, C#) or functions having functionality for communicating with one or more industrial controllers residing in a work environment (*e.g.*, a factory floor). The set of classes include functionality for uploading, downloading, editing and creating of control programs of one or more industrial controllers. Additionally, the set of classes include functionality for accessing control

process data for monitoring and storage of the control process data. Data table values in controller memory can be accessed and edited programmatically through the automation interface 14.

Please replace the 4th paragraph (lines 23-29) on page 11 with the following:

Fig. 5 illustrates a flow diagram of a methodology 100 for implementing downloading to a processor programmatically in accordance with one aspect of the present invention. The methodology begins at 102 where a new application project is created or instantiated, which is opened from a project residing on a disk. Then, at 104, the communication routing to the actual processor that the project represents is set up in the application when the project is opened. This information is saved for later uploading in 106. The project is then downloaded to the processor in 108.

The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 100 of Fig. 5:

```
Public Sub DownloadAProcessor()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.Project
    Dim EncodedRouteData As String
    '** start Project and store it in the g_Application object variable
    Set g_Application = CreateObject("Project.Application")
    '** Declare some variables for clarity.
    Dim ShowDialog As Boolean
    Dim UseAutoSave As Boolean
    Dim IgnorePrompts As Boolean
    Dim OnlineAction As lgxOnlineAction
    '** Initialize these variables to suitable defaults
    ShowDialog = False
    UseAutoSave = False
    IgnorePrompts = True
    OnlineAction = lgxGoOffline
    '** Open a project from disk that will be downloaded.
    Set g_Project  = g_Application.FileOpen( "C:\Projects\Upload.rsp", ShowDialog,
UseAutoSave)
    '** The communication route to the actual processor that this project represents is set up in
    '** the application when the project is opened. Get this info out now and save it away for later
    '**calls to upload this processor.
    EncodedRouteData = g_Project.EncodedRouteString
    SaveEncodedRouteData EncodedRouteData
    '** Now download the image to the processor.
    Dim NoError As Boolean
```

```
    NoError = g_Project.Download(IgnorePrompts, OnlineAction, lgxREMOTEPROG)
End Sub
```

Please replace the 5th paragraph beginning on page 11 (lines 30-31) and ending on page 12 (lines 1-4) with the following:

Fig. 6 illustrates a flow diagram of a methodology 110 for implementing uploading from a processor programmatically in accordance with one aspect of the present invention. At 112, a new application project is created, which initializes routing data. The project is then told which processor to communicate with at 114. Finally, at 116, the function to perform the upload is called and the uploaded program is saved to a file on a disk.

The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 110 of Fig. 6:

```
Public Sub UploadAProcessor()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.Project
    Dim EncodedRouteData As String
    '** start Project and store it in the g_Application object variable
    Set g_Application = CreateObject("Project.Application")
    '** EncodedRouteData holds a string of data that points the project software to the processor
    '** on the Data Highway+ network it will be performing uploads and downloads on. A
    '**function is called here to initialize it to a previously obtained value
    InitializeRouteData EncodedRouteData
    '** Now tell project which processor it is to communicate with
    g_Application.EncodedRouteString = EncodedRouteData
    '** Declare some variables for clarity.
    Dim IgnorePrompts As Boolean
    Dim SaveChanges As Boolean
    Dim AcceptDefaultAction As Booloen
    Dim UploadAction As lgxUploadDownloadAction
    Dim OnlineAction As lgxOnlineAction
    Dim DataBaseAction As lgxSaveAction
    '** Initialize these variables to suitable defaults
    IgnorePrompts = True
    SaveChanges = False
    AcceptDefaultAction = True
    UploadAction = lgxUploadCurrent
    OnlineAction = lgxGoOffline
    DataBaseAction = lgxNoAction
    '** Now make the call that performs the upload. Store the uploaded project object
    Set g_Project = g_Application.Upload(IgnorePrompts,  SaveChanges, UploadAction,
```

```
        _OnlineAction)
    '** Now save the uploaded image to disk as a file called Upload.rsp
    Dim NoError As Boolean
    NoError  = g_Project.SaveAs(IgnorePrompts, AcceptDefaultAction, _
          DataBaseAction, "C:\Projects\Upload.rsp")
End Sub
```

Please replace the 2nd paragraph on page 12 (lines 5-13) with the following:

Fig. 7 illustrates a methodology 120 for inserting ladder logic programmatically in accordance with one aspect of the present invention. At 121, a new application is created, which instantiates a new instance of the automation interface. A project is then opened from disk for modification at 122. Then, at 123, a program file is selected for uploading. The selected program is cast to a ladder file at 124. A sample rung is then built and inserted into the selected program at 125.

The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 120 of Fig. 7:

```
Public Sub InsertLadderRung()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.LogixProject
    Dim g_ProgFile As AutoInterface.ProgramFile
    Dim g_LadderFile As AutoInterface.LadderFile
    '** start AutoInterface and store it in the g_Application object variable
    Set g_Application = CreateObject("AutoInterface.Application")
    '** Declare some variables for clarity.
    Dim ShowDialog As Boolean
    Dim UseAutoSave As Boolean
    Dim IgnorePrompts As Boolean
    Dim AcceptDefaultAction As Boolean
    Dim OnlineAction As lgxOnlineAction
    '** Initialize these variables to suitable defaults
    ShowDialog = False
    UseAutoSave = False
    IgnorePrompts = True
    AcceptDefaultAction = True
    OnlineAction = lgxGoOffline
    '** Open a project from disk that will be modified.
    Set g_Project  = g_Application.FileOpen( "C:\Projects\Upload.rsp", ShowDialog,
          UseAutoSave)
    '** Obtain the program file object for program file 2. Cast this object to a LadderFile object
    '** since in this case, program file 2 is also a ladder file.
    Set g_ProgFile = g_Project.ProgramFiles(2)
```

```
    Set g_LadderFile = g_ProgFile
    '** Build up a sample rung manually
    Dim RungString As String
    RungString = "SOR XIC B3:0/0 OTE B3:0/1 EOR"
    '** Insert this rung into the ladder at position 0
    g_LadderFile.InsertRungAsAscii 0, RungString
    '** Save the modified project to disk first, then
    '** download the image to the processor and set the processor to RUN mode.
    '** Once this is completed, the new rung is executing in the processor.
    Dim NoError As Boolean
    NoError  = g_Project.Save(IgnorePrompts, AcceptDefaultAction)
    NoError  = g_Project.Download(IgnorePrompts, OnlineAction, lgxREMOTERUN)
End Sub
```

It is to be appreciated that the examples of Figs. 5-7 are for illustrated purposes and most error detection/correction code was omitted for the sake of clarity.

REMARKS

The specification has been amended herein to change the descriptions of Figs. 5-7 in accordance with the substitute drawings accompanying the Notice to File Corrected Application Papers, which is being filed concurrently herewith. The specification has also been amended to add information from Figs. 5-7, as originally filed. No new matter has been added. As stated in section 2163.06 of the MPEP, "information contained in any one of the specification, claims or drawings of the application as filed may be added to any other part of the application without introducing new matter." A Clean Version of Replacement Paragraphs to the Specification is found at pages 2-6 of this Reply. A Version With Markings to Show Changes Made is at pages 8-11 of this Reply.

Should there be any questions regarding this paper, the Examiner is invited to contact applicants' undersigned representative at the telephone number listed below.

In the event any fees are due in connection with the filing of this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063.

Respectfully submitted,

Himanshu S. Amin
Reg. No. 40,894

AMIN & TUROCY, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone: (216) 696-8730
Facsimile: (216) 696-8731

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

Please amend the specification as follows:

At page 4, line 7, please replace "code module" with --flow diagram of a methodology--.

At page 4, line 10, please replace "code module" with --flow diagram of a methodology--.

At page 4, line 13, please replace "code module" with --flow diagram of a methodology--.

At page 8, line 2, please replace "unmarhall" with --unmarshall--.

At page 11, lines 23-24, please replace "sample application module 100 or subroutine written in Visual Basic" with --flow diagram of a methodology 100--.

At page 11, line 25, please replace "In line 07," with --The methodology begins at 102 where--

.

At page 11, line 26, please delete "in line 22".

At page 11, line 26, before "The", please add --Then, at 104,--.

At page 11, line 29, please replace "lines 27-28" with --106--.

At page 11, line 29, please replace "lines 31-32" with --108--.

At page 11, line 30, before "Fig. 6", please insert the following paragraph:

-- The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 100 of Fig. 5:

```
Public Sub DownloadAProcessor()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.Project
    Dim EncodedRouteData As String
    '** start Project and store it in the g_Application object variable
    Set g_Application = CreateObject("Project.Application")
    '** Declare some variables for clarity.
    Dim ShowDialog As Boolean
    Dim UseAutoSave As Boolean
    Dim IgnorePrompts As Boolean
    Dim OnlineAction As lgxOnlineAction
    '** Initialize these variables to suitable defaults
    ShowDialog = False
    UseAutoSave = False
    IgnorePrompts = True
```

```
    OnlineAction = lgxGoOffline
    '** Open a project from disk that will be downloaded.
    Set g_Project  = g_Application.FileOpen( "C:\Projects\Upload.rsp", ShowDialog,
UseAutoSave)
    '** The communication route to the actual processor that this project represents is set up in
    '** the application when the project is opened. Get this info out now and save it away for later
    '**calls to upload this processor.
    EncodedRouteData = g_Project.EncodedRouteString
    SaveEncodedRouteData EncodedRouteData
    '** Now download the image to the processor.
    Dim NoError As Boolean
    NoError  = g_Project.Download(IgnorePrompts, OnlineAction, lgxREMOTEPROG)
End Sub--.
```

At page 11, lines 30-31, please replace "sample application module 110 or subroutine written in Visual Basic" with --flow diagram of a methodology 110--.

At page 12, line 1, please replace "In line 09" with --At 112--.

At page 12, line 2, please delete "in line 15".

At page 12, line 3, please replace "in line 18" with --at 114--.

At page 12, line 3, before "The" first occurrence, please add --Finally, at 116,--.

At page 12, line 3, please delete "in line 37".

At page 12, line 4, please delete "at lines 41-43".

At page 12, line 5, before "Fig. 7", please insert the following paragraph:

--The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 110 of Fig. 6:

```
Public Sub UploadAProcessor()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.Project
    Dim EncodedRouteData As String
    '** start Project and store it in the g_Application object variable
    Set g_Application = CreateObject("Project.Application")
    '** EncodedRouteData holds a string of data that points the project software to the processor
    '** on the Data Highway+ network it will be performing uploads and downloads on. A
    '**function is called here to initialize it to a previously obtained value
    InitializeRouteData EncodedRouteData
    '** Now tell project which processor it is to communicate with
    g_Application.EncodedRouteString = EncodedRouteData
    '** Declare some variables for clarity.
```

```
Dim IgnorePrompts As Boolean
Dim SaveChanges As Boolean
Dim AcceptDefaultAction As Booloen
Dim UploadAction As lgxUploadDownloadAction
Dim OnlineAction As lgxOnlineAction
Dim DataBaseAction As lgxSaveAction
'** Initialize these variables to suitable defaults
IgnorePrompts = True
SaveChanges = False
AcceptDefaultAction = True
UploadAction = lgxUploadCurrent
OnlineAction = lgxGoOffline
DataBaseAction = lgxNoAction
'** Now make the call that performs the upload. Store the uploaded project object
Set g_Project = g_Application.Upload(IgnorePrompts,  SaveChanges, UploadAction,
        _OnlineAction)
'** Now save the uploaded image to disk as a file called Upload.rsp
Dim NoError As Boolean
NoError  = g_Project.SaveAs(IgnorePrompts, AcceptDefaultAction, _
        DataBaseAction, "C:\Projects\Upload.rsp")
 End Sub--.
```

At page 12, lines 5-6, please replace "sample application module 120 or subroutine written in Visual Basic" with --flow diagram of a methodology 120--.

At page 12, line 7, please replace "In line 09" with --At 121--.

At page 12, line 9, please replace "line 25 and" with --122. Then, at 123,--.

At page 12, line 9, please delete "at line 29".

At page 12, line 10, please replace "line 30" with --124--.

At page 12, line 10, please delete "in lines 33-34".

At page 12, line 11, please replace "line 37" with --125--.

At page 12, line 11, before "It", please insert the following paragraph:

-- The following is a sample application module or subroutine written in Visual Basic for implementing the methodology 120 of Fig. 7:

```
Public Sub InsertLadderRung()
    Dim g_Application As AutoInterface.Application
    Dim g_Project As AutoInterface.LogixProject
    Dim g_ProgFile As AutoInterface.ProgramFile
    Dim g_LadderFile As AutoInterface.LadderFile
```

```
'** start AutoInterface and store it in the g_Application object variable
Set g_Application = CreateObject("AutoInterface.Application")
'** Declare some variables for clarity.
Dim ShowDialog As Boolean
Dim UseAutoSave As Boolean
Dim IgnorePrompts As Boolean
Dim AcceptDefaultAction As Boolean
Dim OnlineAction As lgxOnlineAction
'** Initialize these variables to suitable defaults
ShowDialog = False
UseAutoSave = False
IgnorePrompts = True
AcceptDefaultAction = True
OnlineAction = lgxGoOffline
'** Open a project from disk that will be modified.
Set g_Project  = g_Application.FileOpen( "C:\Projects\Upload.rsp", ShowDialog,
      UseAutoSave)
'** Obtain the program file object for program file 2. Cast this object to a LadderFile object
'** since in this case, program file 2 is also a ladder file.
Set g_ProgFile = g_Project.ProgramFiles(2)
Set g_LadderFile = g_ProgFile
'** Build up a sample rung manually
Dim RungString As String
RungString = "SOR XIC B3:0/0 OTE B3:0/1 EOR"
'** Insert this rung into the ladder at position 0
g_LadderFile.InsertRungAsAscii 0, RungString
'** Save the modified project to disk first, then
'** download the image to the processor and set the processor to RUN mode.
'** Once this is completed, the new rung is executing in the processor.
Dim NoError As Boolean
NoError  = g_Project.Save(IgnorePrompts, AcceptDefaultAction)
NoError  = g_Project.Download(IgnorePrompts, OnlineAction, lgxREMOTERUN)
End Sub--.
```